THE NATURE OF COMPUTATION:

AN INTRODUCTION TO COMPUTER SCIENCE

Ira Pohl Alan Shaw

THE NATURE OF COMPUTATION:

AN INTRODUCTION TO COMPUTER SCIENCE

Ira Pohl
University of California at Santa Cruz

Alan Shaw
University of Washington

COMPUTER SCIENCE PRESS

Computer Science Press, Inc.
11 Taft Court
Rockville, Maryland 20850

1 2 3 4 5 6 86 85 84 83 82 81

Library of Congress Cataloging in Publication Data

Pohl, Ira.
The nature of computation.

(Computer software engineering series)
Bibliography: p.
1. Electronic data processing. I. Shaw, Alan C., 1937- joint author. II. Title. III. Series.
QA76.P58 001.64 80-18433
ISBN 0-914894-12-9

Cover design by Ruth Ramminger

PREFACE

This book is a rigorous introduction to computer science, suitable for the beginning major. As a general introduction, it has three objectives: to provide a survey of the field; to provide an initial literacy in the language and methods of computer science; and to present an historical, philosophical, and social perspective. The principal subjects treated are algorithms and their description, data and its representation, Boolean algebra, computer circuits and organization, programming languages and systems, computability theory, analysis of algorithms, and applications and implications of computing.

Our primary theme is that computer science is the study of algorithms. The technical foundations and applications are covered chiefly by constructing and studying a number of basic algorithms for each topic. Examples are Euclid's algorithm, number conversion, computer arithmetic, text manipulation, various machine simulations, elementary Turing machines, polynomial evaluation, sorting, and binary search. The purpose is not only to understand and apply the algorithms but also ultimately to analyze their complexity and prove their correctness. In order to describe algorithms, we define a small Algol-like programming notation, dubbed "Algolic," that most closely resembles a subset of Pascal.

Because computer science and technology are intimately tied to a societal context, we discuss the history, applications, and social implications of various technical developments. Significant contemporary work as well as the contributions of pioneers such as Babbage, Turing, and von Neumann are described. Also introduced are some of the philosophical, moral, and social controversies surrounding advances in computing, including those related to non-computability, artificial intelligence, computer modelling, and data banks and privacy.

Programming is introduced in the book through many examples and exercises. The progression of Algolic and examples is constructed to allow a parallel course in programming; alternatively, a programming course may follow our first course. Both approaches have been used successfully.

The present text evolved from a set of notes originally developed by I. Pohl in 1971. Early drafts of this book have been used by the authors as the text for the basic introductory course in computer science at the University of California at Santa Cruz since 1971 and at the University of Washington since 1975. After some experimentation, we have found that a one-term course in calculus provides the minimum mathematical sophistication for the material. The entire book can be covered in a year-long course, or selected parts can be used for a single term offering.

Contents

Preface vii

1. COMPUTER SCIENCE 1

1.1 Origins of Computer Science 1
1.2 On Defining Computer Science 3
1.3 Core Topics: Technical, Applications, and Social 4
1.4 Outline of This Text 7
Exercises 8
Additional Readings 8

2. ALGORITHMS 9

2.1 Algorithms, Programs, and Computations 9
2.2 Description and Analysis of Algorithms 16
2.2.1 Euclid's Algorithm 16
2.2.2 Correctness 20
2.2.3 Program Descriptions 22
2.3 Performing Algorithms on Computers 26
2.4 Introduction to the Algolic Language 34
2.4.1 Variables and Expressions 35
2.4.2 Statements 37
2.4.3 Examples of Algolic Programs 41
Exercises 48
Additional Readings 50

3. NUMBERS AND OTHER DATA 51

3.1 The Origins of the Number System 51
3.1.1 Ancient Civilizations and Their Number Systems 52
3.1.2 The Zero: Can Nothing Be Counted? 56
3.1.3 The Abacus 58
3.1.4 Ancient and Medieval Algorithms 60
3.2 Radix Positional Notation 63
3.2.1 Decimal, Binary, and Other Systems 63

3.2.2 Iteration and Arrays in Algolic 68
3.2.3 Converting Between Bases 74
3.3 Coding the Integers 79
3.4 Floating Point Numbers 85
3.5 Characters and Strings 89
3.6 Data and Structure 93
Exercises 97
Additional Readings 99

4. BOOLEAN ALGEBRA AND APPLICATIONS 100

4.1 Propositional Logic 101
4.1.1 Elements of Propositional Logic 102
4.1.2 Some Applications of Logic 106
4.2 Boolean Expressions in Programming Language 107
4.3 Boolean Algebra 111
4.4 Switching Algebra and Combinational Circuits 119
4.4.1 Completeness 123
Exercises 125
Additional Readings 129

5. COMPUTER ORGANIZATION AND DESIGN 130

5.1 A Short History of Digital Computers 130
5.1.1 Charles Babbage 132
5.1.2 The Modern Computer: Contemporary Developments 137
5.2 Hardware Structure and Building Blocks 142
5.3 Basic Electronic Components and Circuits 144
5.4 Digital Logic 152
5.4.1 Combinational Networks 152
5.4.2 Storage Systems 157
5.4.3 Sequential Circuits 162
5.5 Computer Architecture: An Example Machine 171
5.5.1 Programmer's Specification 172
5.5.2 Data Flow and Instruction Processing 177
5.5.3 Extensions and Variations 181
5.6 Computer Systems and Technology 189
Exercises 190
Additional Readings 192

6. PROGRAMMING LANGUAGES AND SYSTEMS 193

6.1 Software Components of Computer Systems 194
6.2 Programming Languages 201
6.2.1 Historical Perspectives 201
6.2.2 Syntax Specification 205
6.2.3 Data Types and Structured Data 213
6.2.3.1 Arrays 216
6.2.3.2 Records 219
6.2.3.3 Dynamic Structures 221
6.2.4 Procedures and Functions 224
6.3 Compilers 227
6.3.1 Polish Postfix Notation 230
6.3.2 Symbol Tables 237
6.4 Operating Systems 242
6.4.1 OS Architecture 244
6.4.2 Interactions Among Concurrent Processes 249
Exercises 256
Additional Readings 260

7. TURING MACHINES AND COMPUTABILITY 261

7.1 Turing Machines and the Church-Markov-Turing Thesis 261
7.2 Some Examples of TM Computations 266
7.3 TM Composition 274
7.4 The Universal Turing Machine 275
7.5 The Halting Problem 277
7.6 Numbers, Infinity and Diagonalization: Analogues to the Halting Problem 280
7.7 Variants on the Halting Problem 284
7.8 Alternative Forms of Computability 285
7.9 The Significance of Non-Computability 287
Exercises 288
Additional Readings 292

8. ANALYSIS OF ALGORITHMS 294

8.1 Program Proving 295
8.2 Efficiency 302
8.3 The Efficiency of Searching and Sorting 305

8.4 What is Practically Computable 310
Exercises 311
Additional Readings 314

9. ARTIFICIAL INTELLIGENCE 315

9.1 Turing's Test and Definitions of AI 316
9.2 Game Playing 318
9.2.1 A Detailed Look at Playing Chess 324
9.3 Heuristic Problem Solving 327
9.4 Natural Language and Semantics 333
9.5 Learning 337
9.6 Artificial Intelligence: Trends, Achievements, and Problems 340
Exercises 344
Additional Readings 346

10. COMPUTER USE AND ITS IMPLICATIONS 347

10.1 Social and Ethical Questions 347
10.1.1 An Example of the Thin Gray Line of the Social Consequences of Technology 348
10.2 Computers in Medicine 350
10.3 Computers and Decision Making 353
10.4 Information Retrieval and Privacy 362
10.5 Forecasts 367
10.6 Technology and Technocracy: The Challenge to Man's Values 369
Exercises 371
Additional Readings 371

APPENDIX 373

INDEX 387

Chapter 1

COMPUTER SCIENCE

1.1 ORIGINS OF COMPUTER SCIENCE

Revolutionary advances in both science and engineering have changed the content and methodology of almost all technical fields during the current century, including physics, biology, medicine, electrical engineering, chemistry, transportation, communication, textiles, energy, agriculture, and computing. Most of the advances and changes have been incorporated quite naturally within traditional subject areas, and rarely has a major new discipline been created. Computer science is one of these rarities.

While computer science is undoubtedly the youngest of the sciences, its origins can be traced back many years. The most significant events of recent historical interest were the constructions of the first digital computers during and immediately after World War II. In the sense that the *science of X is the study of X*, computer science is devoted to the study of those devices called computers; the field could not exist until the first machines were built in the 1940's. However, hardware ideas about computers were expounded much earlier. The most prominent example is the work of C. Babbage who derived the basic principles of computer organization and designed a machine in the early 19th century.

Much of the theoretical foundations of computer science started with the research of A. Turing in the 1930's. He developed a mathematical model of a universal computing device and proved some remarkable properties of this abstract device. Turing's "machine" is still the accepted theoretical model of a computer. In summary, the modern origins of the field are the construction of the first computers and the independent formulation of an adequate mathematical model of machines.

The subject was recognized as a separate academic discipline in the mid 1960's when computer science departments were created at a number of universities in the U.S.A., and graduate programs, usually leading to the Master of Science and Doctor of Philosophy degrees, were introduced. In 1968, the Association for Computing Machinery, the largest professional

and scientific organization of computer scientists, published a model undergraduate curriculum; this became the basis for many college and university programs throughout the world. It did not take long for computer science to become established, with all the trappings of a traditional discipline such as many technical journals and conferences, academic departments at most institutions of higher education, scientific and professional associations, and thriving research laboratories in academia, industry, and government.

Why is computer science a separate major field? After all, a computer is *just* a grand calculating device. Previous calculators, such as slide rules, were only considered as computing tools and never seriously treated as a new basic area of knowledge. Other significant technical innovations such as automobiles, television, and lasers have not led to separate disciplines; instead they just became part of the existing fields of mechanical engineering, electrical engineering, and physics, respectively. At one level, it is difficult to answer our question without assuming some familiarity with this book. However, we can make two main points that should be comprehensible now and can be profitably reread later.

Our first point is a technical one, dealing with the content of the field. Surprisingly, *computers are universal problem solving machines.* Any procedure that is precisely formulated can be implemented on a computer; such computational procedures are called *algorithms*. As a consequence of this universality, the computer has permitted scientists to focus on all aspects of technical problem solving. These include languages and notations for describing problems and their solutions, the efficiency and optimality of various problem solving schemes, the classes of computing devices required for different types of problems, the inherent complexity of specific problems, the existence or non-existence of solutions to well-posed problems, and a host of other related questions. Like mathematics, computer science provides the necessary tools for many technical endeavors. Unlike mathematics and similar to engineering and the natural or social sciences, computer science also has the very practical goals of understanding and applying real world phenomena—those surrounding computers and algorithmic problem solving.

The second point that accounts for the creation of a computer science is social. The impact of computers on society has been enormous but not well understood. It is evident that the computer "revolution" is still in progress. The ultimate effects on our political, cultural, economic, and moral lives are unpredictable, other than acknowledging that they will be substantial and potentially range from very beneficial to very hazardous. Since much of our economy relies on automatic information processing, there is a need

for people who are technically trained in computers. At a different level, it is recognized that a new computer ingredient has been introduced in our societies, and that every well-educated person should have a basic understanding of these machines and some of their implications.

1.2 ON DEFINING COMPUTER SCIENCE

The science of any subject is concerned with the accumulation and organization of information about the subject and with the derivation and discovery of principles and methods. Most sciences have a pure aspect that is devoted entirely to the expansion of knowledge for its own sake, and an applied component that is oriented towards using this knowledge. The range covered by computer science is very broad. At one end of the spectrum, we have theoretical, and sometimes philosophical, investigations on the ultimate capabilities of machines and on the properties of various general problems and algorithms; the more applied end deals with techniques for the design and construction of computer systems, i.e. computer engineering, and with advanced applications.

The computer scientist, in his applied role, has been characterized as a *toolmaker*. As such, he develops methods and machines that others use in solving their problems. The theoretical computer scientist, while also involved in toolmaking, studies the nature of the tool. In a natural science, some part of the given natural or physical world is investigated. Computer science, by contrast, is an *artificial science* since we are studying one of man's own creations. As a result, the interest is not so much in observed facts and the discovery of laws, as it is in methods for the analysis and design of these creations and the derivation of their properties.

Computer science has been defined in many different ways. Following are three definitions that have been proposed.

1. Device-centered
   This is the straightforward definition that was mentioned in the last section.
   "Computer science is the study of computers."
   Thus the science focuses on the device and on questions surrounding it, such as how to design, analyze, construct, and use computers.
2. Information-centered
   Information or data is considered the central notion here.
   "Computer science is the study of information."
   This leads to questions about the representation, storage, organization, transmission, and processing of data.

3. Program-centered
   A *program* is a set of computer instructions for solving a specific problem. Computer programs are also referred to as *software*.
   "Computer science is the study of programming."
   In this view, the emphasis is on programming issues such as appropriate languages for expressing programs, the correctness and efficiency of programs, machines for executing programs, and programming techniques.

All these definitions are "correct" in that most computer science topics are encompassed by any of them. The first definition emphasizes real and abstract machines, the second the data that machines work with, and the third the programs used to communicate with machines. Because their emphases are different, they are subject to misinterpretation and can easily be construed as too narrow or too broad.

What is desired is a definition that gracefully includes all of the above views without unduly emphasizing any. The unifying notion of an algorithm, informally defined in the last section, satisfies this requirement.

4. Algorithm-centered
   "Computer science is the study of algorithms."
   Computers are machines for implementing algorithms, information is the "stuff" that algorithms manipulate and produce, and programming is the means for describing algorithms. The algorithm concept also easily includes the theoretical parts of computer science.

Algorithms are the theme of this book and we shall see that all computer science questions can be naturally formulated as questions about algorithms.

1.3 CORE TOPICS: TECHNICAL, APPLICATIONS, AND SOCIAL

The theory and practice of computer science covers technical, applications, and social issues. The relations among these three aspects of the field are illustrated in Figure 1.1. Thus technical advances lead to applications and social questions, and, in turn, the technical side of computing is strongly influenced by applications and social pressures. Similarly, social problems feed back to applications development and technical questions. While concerned primarily with scientific or technical problems, the computer scientist must be aware of all three contexts of his work.

In this section, we introduce some of the main computer science topics in the above three areas. A summary of the core topics is presented in

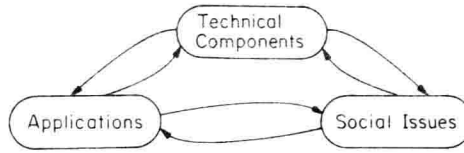

Figure 1.1 Relations Among computer Science Topics

Table 1.1. Our list should not be considered static but one that is constantly expanding as the field progresses and new insights are made.

The technical component can be summarized as the design, implementation, analysis, and theory of algorithms. A computer scientist attempts to answer some of the following questions:

1. What is an appropriate machine design for efficiently executing algorithms?
2. What is a good programming language for writing algorithms?
3. How can one design and implement good algorithms?
4. Is a particular algorithm correct for all possible cases?
5. How efficient is a particular algorithm?
6. What is the best possible algorithm for a given problem?
7. Does there exist an algorithm to solve a particular problem?
8. How does one define the syntax (form) and semantics (meaning) of a programming language?

Computer science research takes place within the framework of certain applications that are a source of computing problems of general interest. Scientific computing, which deals with algorithms for the numerical solution of mathematical equations, was the original application of computers, with a history stretching far back to antiquity and the earliest calculating devices. Another source of, usually numeric, algorithms is in problem solving by simulation; real world predictions are attempted by creating and executing a computer model of the phenomenon being studied. Investigations of nonnumeric algorithms, algorithms that manipulate symbols rather than numbers, arise naturally in the fields of artificial intelligence and database systems. The goal of artificial intelligence research is to produce computer systems that exhibit "intelligent" behavior; this ambitious, and often controversial, goal has generated a rich set of problems in game-playing, symbolic mathematics, theorem proving, and natural language understanding. The last application in our list, database systems,

Table 1.1 Some Core Topics of Computer Science

A. Technical Components
1. Design: machines, programming languages and systems, programs
2. Analysis: correctness, efficiency
3. Theory and Foundations: automata theory and computability, complexity, formal languages
B. General Applications
1. Scientific Computing
2. Modelling and Simulation
3. Artificial Intelligence
4. Database Systems
C. Social Issues
1. Machine Intelligence
2. Automation
3. Privacy
4. Planning with Computer Models

refers to large information storage and retrieval systems; examples are systems for libraries, airline reservations, automobile design, income tax, credit, and banking.

At least four related social issues are direct and serious consequences of computer science developments. The possibilities and implications of machine intelligence are the subject of countless popular and scholarly discussions; these may stress, for example, the assault on man's ego caused by intelligent machines, the benefits of a benevolent intellectual collaboration between men and machines, or the horrors of being ruled by computers. As more and more tasks become automated and delegated to computers, society must deal with the undesirable effects such as possible widespread unemployment and the desirable aspects such as increased leisure time. The existence of large database systems containing detailed personal data has already caused great concern about the loss of privacy and potential for misuse; the prospects of "big brother" keeping track of one's every movement and transaction appeal to few, yet these systems also produce considerable economic and social benefits. The last issue stems from the increased reliance on computer modelling for economic and social planning. Large computer models give planning agencies much better information for intelligent decision-making. On the other hand, some simulations and models are so complex that few understand the underlying assumptions in the model and can interpret the results sensibly; the danger is that the computer output of such models is accepted and used on faith alone.

The nuclear and biological sciences have painfully learned that science is not value free and that the social implications of research directions and

advances must be considered. Computer science is also fundamentally tied to its societal context and the relationships between the science and society cannot be ignored.

1.4 OUTLINE OF THIS TEXT

The chapters and topics in our book are sketched in Figure 1.2. The arrows in the figure indicate prerequisites.

The introductory part in the next three chapters presents some basic material on algorithms, data, and Boolean algebra. Following this are the central technical subjects: machines, software, theory of computation, and analysis of algorithms. The last two chapters deal with applications and the societal implications of computer technology and science. In order to give a deeper feeling for the dynamics of computer science—its creation and evolution—and also to acknowledge the major contributing pioneers, we also discuss the history of each development.

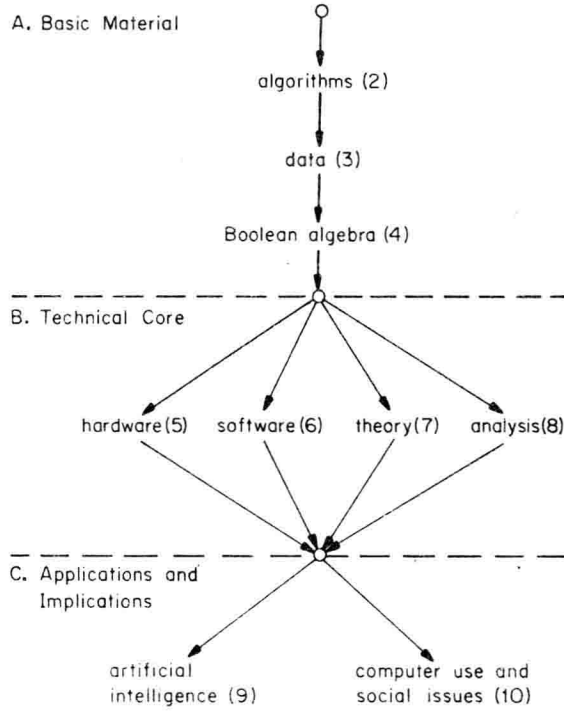

Figure 1.2 Book Outline

The technical foundations are covered chiefly by constructing and studying a number of basic algorithms relevant to each topic. To describe algorithms, we use a precise language called *Algolic*. Algolic is defined informally in the next few chapters; a more formal specification and summary is given in the Appendix.

The end of each chapter has an Exercise section and a list of references for additional reading.

EXERCISES

1. Make a list of all the interactions you have with computers, e.g., utility bills, automatic mailing lists, class scheduling, ...

2. Discuss the following two statements:
   (a) "The computer, like a hammer, is only a tool that may be applied to study other disciplines. Therefore, there is no compute science."
   (b) The study of computers is just a fad similar to the earlier fad involving the study of automobiles."
   (Note: It is possible to agree with either of these two statements.)

3. Give a detailed step-by-step procedure, i.e., an "algorithm," for each of the following tasks or problems:
   (a) travelling from your place of residence to school or work
   (b) manually dividing one number by another (long division)
   (c) taking an unordered set of numbers and putting them in ascending sequence, i.e., "sorting" the numbers in ascending sequence
   (d) playing, and never losing, the game of tic-tac-toe.

ADDITIONAL READINGS

Newell, A.; Perlis, A. J.; and Simon, H. A. "Computer Science." *Science,* vol. 157, 1 September 1967, 1373-1374.

Phol, I., and Shaw, A. "Introducing Computer Science: An Alternative." *Proc. IFIP 77,* North-Holland, Amsterdam, 1977, pp. 53-56.

Chapter 2

ALGORITHMS

Our principal theme is that computer science is the study of algorithms. The field is centrally concerned with such topics as the theory and properties of algorithms, methods for the design and analysis of algorithms, programming and other languages for describing algorithms, systems and techniques for representing algorithms as programs, machines for executing algorithms, and the societal implications of computer science and technology. In this chapter, we discuss the concept and characteristics of an algorithm, introduce several notations for expressing algorithms including the language *Algolic* used throughout the book, and define a simple computer than can execute algorithms. The material is presented mainly by constructing and studying a variety of example algorithms.

2.1 ALGORITHMS, PROGRAMS, AND COMPUTATIONS

Informally, an *algorithm* is a list of instructions for performing a specific task or for solving a particular type of problem. Algorithm-like specifications for problem solving and task performance are commonly found in our everyday experiences. Examples include the detailed instructions for knitting a sweater, making a dress, cooking a favorite meal, registering for classes at a university, travelling from one destination to another, and using a vending machine. It is instructive to examine one of these examples.

Consider the following recipe for preparing a meat roast:

Sprinkle the roast with salt and pepper. Insert a meat thermometer and place in oven preheated to 150°C. Cook until the thermometer registers 80°C-85°C. Serve roast with gravy prepared from either meat stock or from pan drippings if there is a sufficient amount.

The recipe is typically imprecise—what does "sprinkle" mean, where is the thermometer to be inserted, what is a "sufficient amount" of pan drippings?